UNIVERSIDAD DISTRITAL FRANCISCO JOSÉ DE CALDAS
Acreditación Institucional de Alta Calidad

TIA
Tecnología, Investigación y Academia

Arquitectura basada en micro-servicios para aplicaciones web

Microservices based architecture for web applications

Ángela Indira Rodríguez[1], Jackelyne Ivone Padilla[2], Hugo Alexander Parra[3].

Para citar: Rodríguez, Á., Padilla, J., Parra, A (2019), Arquitectura basada en micro-servicios para aplicaciones web. TIA,7(2), pp. 12-xxxxxx.

Resumen:

Los grandes avances tecnológicos que se presentan día a día están cambiando la forma en cómo se interactúa con el mundo en el diario vivir. Desde las aplicaciones web hasta las móviles están siendo modificadas y enfocadas al consumo de servicios. Estas adaptaciones acarrean grandes retos para las aplicaciones que están actualmente diseñadas bajo arquitecturas antiguas, en donde la funcionalidad y sus procesos son dependientes entre sí, haciendo que los cambios sean sumamente tediosos, costosos y con alta probabilidad de fallo.

Para estos casos, se hace necesario el uso de una arquitectura que permita crear aplicaciones escalables, flexibles, con independencia en sus funciones y en donde igualmente permita adaptar las aplicaciones ya existentes enfocadas a los servicios.

Esta arquitectura es la basada en micro-servicios, en donde posee grandes ventajas en varios ámbitos a la hora de desarrollar aplicaciones web.

Palabras Clave: Arquitectura, software, microservicios, aplicaciones web

Artículo de investigación

Fecha de recepción:
2018-05-23
Fecha de recepción:
2018-12-13

ISSN: 2344-8288
Vol. 7 No. 2
Julio- diciembre 2019
Bogotá-Colombia

Abstract:

In this research, you can find information about Augmented Reality, the advances it has had the great technological advances that occur every day are changing the way we interact with the world in our daily lives. From web to mobile applications are being modified and focused on the consumption of services. These adaptations bring great challenges to applications that are currently designed under old architectures, where functionality and processes are dependent on each other, making the changes extremely tedious, costly and with high probability of failure.

For these cases, it is necessary to use an architecture that allows the creation of scalable and flexible applications, with independence in their functions and where it is possible to adapt existing applications focused on services.

This architecture is based on micro-services, where it has great advantages in several areas when developing web applications.

Key Word: Architecture, software, microservices, web applications

[1] Ingeniera telemática, Universidad Distrital Francisco Jose de Caldas, anirodriguez02@gmail.com, Assist, Bogotá.
[2] Ingeniera sistemas, Universidad Distrital Francisco Jose de Caldas,Jackpave0917@gmail.com, BBVA Fiduciaria, Bogotá
[3] Ingeniero electrónico, Universidad Distrital Francisco Jose de Caldas,hugparra88@gmail.com, BBVA Seguros, Bogotá

I.Introducción

Durante los años 2014 y 2015 las empresas como Netflix y Amazon se han desempeñado en realizar software teniendo casos de éxitos implementando arquitecturas basadas en microservicios. Por ejemplo Amazon, desde el año 1994 nació con la filosofía muy similar que hoy se conoce como microservicios , desde el inicio entendieron que los equipos pequeños pueden trabajar de forma más óptima que los equipos grandes, por ello y siendo una empresa con la capacidad para generar las suficientes herramientas para apoyar a sus desarrolladores, crearon los Amazon Web Services, permitiendo a sus equipos encargarse de todo el ciclo de vida de los proyectos, además permitiéndoles contar con infraestructura diseñada especialmente para sus proyectos [1].

Netflix por medio de su sitio web, en el cual son muy activos sus arquitectos, se ha convertido en las principales referencias de libros como: Building Microservices y artículos como Microservices en donde se ejemplifican los diversos patrones implementados por la empresa. Entre estos patrones, uno de los más comentados es su API Gateway. Además, por medio de su Open Source Software Center proveen muchas de sus herramientas de forma gratuita para los desarrolladores en general [2]

2.Microservicios

Como lo definen James Lewis y Martin Fowler en su artículo: *Microservices*, uno de los artículos más populares sobre el tema en la web, “es una forma particular de diseño de aplicaciones de software como suites de servicios desplegados independientemente” [3], la arquitectura de Microservicios es un enfoque de desarrollo de una aplicación como un conjunto de servicios más pequeños, cada uno se ejecuta en su propio proceso y se exponen, normalmente, a través de protocolos HTTP mediante APIs RESTFull. Estos servicios están construidos alrededor de capacidades o funcionalidades de negocio y con independencia de despliegue a través de un sistema de automatización. Además, pueden estar escritos en diferentes lenguajes de programación y utilizar diferentes tecnologías de almacenamiento de base de datos [4].

2.1 Características del Software de microservicios

Aunque no se encuentra un estándar basado en la arquitectura de servicios, se puede resaltar las siguientes características:

- El software construido en base a microservicios se podrá descomponer en varias partes funcionales independientes. Es decir, su programación o desarrollo es para que así mismo sea más eficiente al implementar algún requerimiento nuevo, evitando redespliegues de toda una aplicación.
- Su organización está basada en torno a las necesidades, capacidades y prioridades del cliente o negocio en el que se implantará, usando módulos multifuncionales, adaptando un módulo común que ofrezca un servicio determinado, generando así ahorro de tiempo en desarrollos.
- El funcionamiento del software de microservicios puede parecerse al sistema de trabajo clásico de UNIX

donde se recibe la petición, se procesa y se genera una respuesta**.**

- La arquitectura de microservicios mantiene un sistema similar a un gobierno descentralizado, donde cada módulo contará por ejemplo con su propia base de datos, en lugar de acudir todos a la misma sobre cargándola así de solicitudes y arriesgándose a que si falla ésta, toda la aplicación caiga.

- Cuando varios servicios están comunicados entre sí, por lo general contarán con un sistema de aviso y actuación si alguno de estos servicios llega a fallar (como mostrar una advertencia, enviar un mail a soporte, avisar a los usuarios de un fallo temporal, etc.…), filtrando adecuadamente la información destinada a este módulo y favoreciendo la correcta gestión de los recursos entre los módulos funcionales restantes [5].

Ventajas

- Equipo de trabajo mínimo
- Escalabilidad
- Funcionalidad modular, módulos independientes.
- Libertad del desarrollador de desarrollar y desplegar servicios de forma independiente
- Uso de contenedores permitiendo el despliegue y el desarrollo de la aplicación rápidamente.

Desventajas

- Alto consumo de memoria
- Necesidad de tiempo para poder fragmentar distintos microservicios
- Complejidad de gestión de un gran número de servicios
- Necesidad de desarrolladores para la solución de problemas como latencia en la red o balanceo de cargas
- Pruebas o testeos complicados al despliegue distribuido [6].

2.3 Arquitecturas basadas en Microservicios

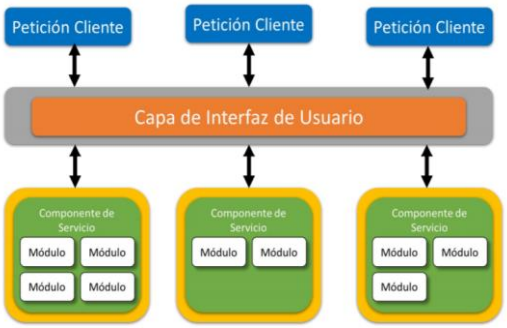

Fig. 1. Patrón Básico de Arquitectura de Microservicios [7]

La arquitectura de microservicios ha sido el término de moda en los últimos tiempos, pero la idea no es nueva en absoluto. De hecho, es bastante similar al patrón SOA que fue muy popular hace unos años [8]. Tanto los microservicios como SOA se tratan de descomponer aplicaciones en servicios más pequeños para escalamiento y manejo de ciclos de vida más eficientes. Sin embargo, no son iguales [9]. Mientras que SOA es un concepto general y puede significar muchas cosas, la arquitectura de microservicios describe un modo particular de construir aplicaciones con servicios muy pequeños, cada uno de los cuales se enfoca en hacer sólo una cosa bien [10]. Hay dos razones por las cuales los microservicios están ganando popularidad.

- En primer lugar, como los aplicativos son cada vez más complejos, la arquitectura monolítica tradicional ya no cumple con las necesidades de escalabilidad y ciclo de desarrollo rápido (en la siguiente sección veremos cómo los microservicios lo hacen posible) [11].
- En segundo lugar, el éxito de grandes compañías de Internet, principalmente

Netflix, al implementar la arquitectura de microservicios, es una gran motivación para que otras empresas consideren hacer el cambio [12].

2.4 Arquitecturas Monolítica

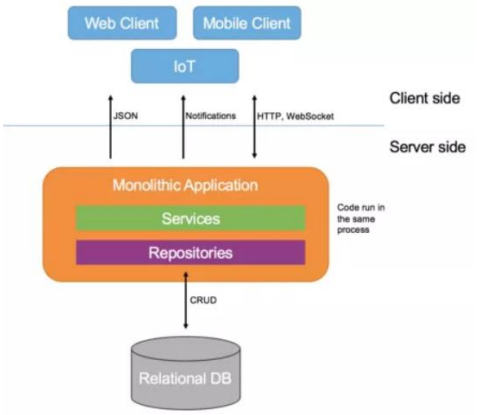

Fig. 2. Arquitectura Monolítica [13]

En la arquitectura de microservicios, la aplicación monolítica se descompone en múltiples servicios pequeños, granulares, aislados**,** independientes y distribuibles [14]**.** El hecho de que estos servicios se desacoplen por separado es trascendental, pues permite priorizar los recursos escasos a los microservicios más relevantes, sobre los demás [15].

2.5 Arquitecturas Microservicios

Una arquitectura de microservicios consta de una colección de servicios autónomos y pequeños. Los servicios son independientes entre sí y cada uno debe implementar una funcionalidad de negocio individual [16].

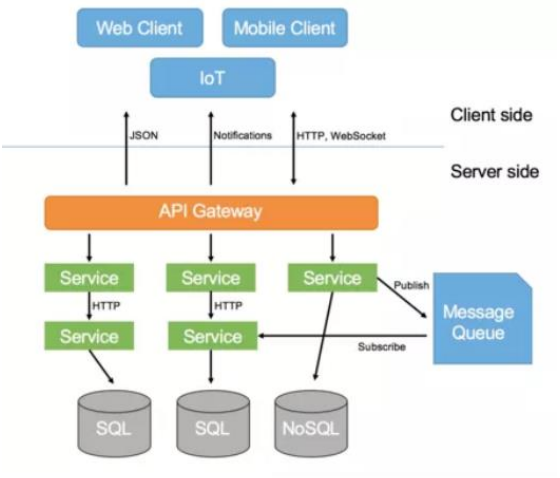

Fig. 3. Arquitectura Microservicios [17]

2.6 Características de la Arquitectura de Microservicios

En una arquitectura de microservicios, los servicios son pequeños e independientes y están acoplados de forma flexible.
Cada servicio es un código base independiente, que puede administrarse por un equipo de desarrollo pequeño.

- Los servicios pueden implementarse de manera independiente. Un equipo puede actualizar un servicio existente sin tener que volver a generar e implementar toda la aplicación.
- Los servicios son los responsables de conservar sus propios datos o estado externo. Esto difiere del modelo tradicional, donde una capa de datos independiente controla la persistencia de los datos.
- Los servicios se comunican entre sí mediante API bien definidas. Los detalles de la implementación interna de cada servicio se ocultan frente a otros servicios.
- No es necesario que los servicios compartan la misma pila de tecnología, las bibliotecas o los marcos de trabajo [18].

2.7Monolítica vs Microservicios

Tabla 1. Arquitecturas monolíticas vs. microservicios [9]

Categoría	Arquitectura Monolítica	Arquitectura de microservicios
Código	Una base de código única para toda la aplicación.	Múltiples bases de código. Cada microservicio tiene su propia base de código.
Comprensibilidad	A menudo confuso y difícil de mantener.	Mayor facilidad de lectura y mucho más fácil de mantener.
Despliegue	Implementaciones complejas con ventanas de mantenimiento y paradas programadas.	Despliegue sencillo ya que cada microservicio se puede implementar de forma individual, con un tiempo de inactividad mínimo, si no es cero.
Lenguaje	Típicamente totalmente desarrollado en un lenguaje de programación.	Cada microservicio puede desarrollarse en un lenguaje de programación diferente.
Escalamiento	Requiere escalar la aplicación entera, aunque los cuellos de botella estén localizados.	Cada microservicio puede desarrollarse en un lenguaje de programación diferente.

Fig. 4. Arquitectura Monolítica vs Microservicios [19].

2.7.1Comunicación entre Microservicios

En una aplicación monolítica los componentes se invocan uno a otro con llamadas de funciones. Una aplicación basada en microservicios es un sistema distribuido en múltiples máquinas [20]. Cada instancia del servicio es típicamente un proceso o una máquina, luego hay que aplicar mecanismo de comunicación entre procesos [21].

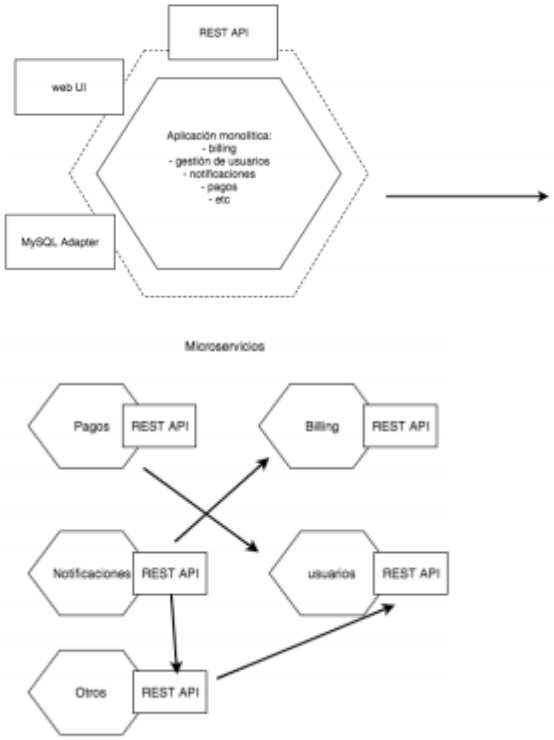

Fig. 5. Comunicación Microservicios [22]

Una aplicación basada en microservicios es un sistema distribuido que se ejecuta en varios procesos o servicios, normalmente incluso en varios servidores o hosts. Lo habitual es que cada instancia de servicio sea un proceso. Por lo tanto, los servicios deben interactuar mediante un protocolo de comunicación entre procesos como HTTP, AMQP o un protocolo binario como TCP, en función de la naturaleza de cada servicio normalmente los microservicios que componen una aplicación de un extremo a otro se establecen sencillamente mediante comunicaciones de REST en lugar de protocolos complejos como WS-* y comunicaciones flexibles controladas por eventos en lugar de orquestadores de procesos de negocios centralizados [23].

Los dos protocolos que se usan habitualmente son respuesta-solicitud HTTP con API de recurso (sobre todo al

consultar) y mensajería asincrónica ligera al comunicar actualizaciones en varios microservicios [24].

2.7.2 Tipos de comunicación

Colas de mensajes: son almacenamientos temporales para mensajes que necesitan compartirse de un sistema a otro, esta comunicación funciona de forma asíncrona; teniendo, por un lado, al cliente que inserta el mensaje a la cola; por el otro. Entre los sistemas de colas de mensajes se puede encontrar a RabbitMQ, Apache ActiveMQ, entre otros.

Servicios REST: La transferencia de estado se refería a un conjunto de principios de arquitectura, pero hoy en día ha cambiado para describir cualquier interfaz entre sistemas que utilice directamente HTTP utilizando diferentes formatos como JSON, XML, entre otros.

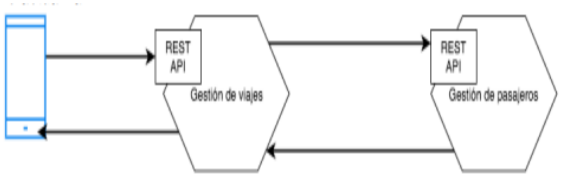

Fig. 6. Comunicación Microservicios [25]

Node.js: Es un entorno en tiempo de ejecución multiplataforma construido sobre el motor JavaScript versión 8. Maneja un modelo de no bloqueo y manejo por eventos que lo hacen ligero y eficiente. Node.js permite desarrollar bajo el paradigma de programación reactiva por medio de la funcionalidad EventEmitter que permite la suscripción a eventos y actuar sobre ellos de forma asíncrona.

Una aplicación basada en microservicio suele usar una combinación de estos estilos de comunicación. El tipo más común es la comunicación de un único receptor con un protocolo sincrónico como HTTP/HTTPS al invocar a un servicio normal HTTP Web API. Además, los microservicios suelen usar protocolos de mensajería para la comunicación asincrónica entre microservicios [26].

Mensajería sincrónica frente a la asincrónica. Hay dos patrones de mensajería básicos que los microservicios pueden usar para comunicarse con otros microservicios.

Comunicación sincrónica: En este patrón, un servicio llama a una API que otro servicio expone mediante un protocolo como HTTP o gRPC. Esta opción es un patrón de mensajería sincrónico porque el autor de la llamada espera a la respuesta del receptor.

Paso de mensajes asincrónicos: En este patrón, un servicio envía el mensaje sin esperar por la respuesta, y uno o más servicios procesan el mensaje de forma asincrónica [27].

2.8 Modelado de microservicios

Los microservicios poseen una estructura interna similar y se componen de las capas de protocolo, dominio, externa, persistencia como se muestra a continuación.

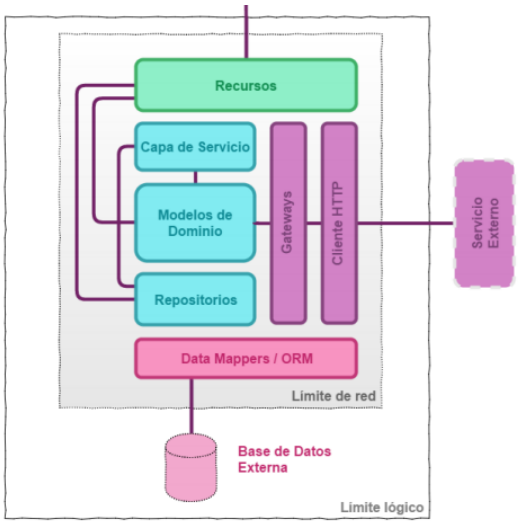

Fig. 7. Modelado de Microservicios [28]

El protocolo está expuesto por el servicio y los objetos representados por el dominio. lo recursos son livianos y verifican las peticiones a la red y obtener una respuesta al protocolo de acuerdo con el resultado de la petición del negocio. Por otro la capa de servicio representa la lógica del negocio lo que se denomina dominio, la persistencia en un microservicio persiste los objetos del dominio entre las peticiones y el externo si un servicio tiene otro servicio, será fundamental la comunicación con el servicio externo, una parte encapsula el mensaje transformando peticiones y respuestas con los objetos del dominio [29].

3. Referencias

[1] Richardson, c. (2014). Microservices: decomposing applications for deployability and scalability. Infoq. http://www.repositorio.usac.edu.gt/7023/1/jos%c3%89%20manuel%20de%20paz%20estrada.pdf

[2] Balalaie, a., Heydarnoori, a. y Jamshidi, p. (2015). Migrating to cloudnative architectures using microservices: an experience report. Sharif University of technology, Tehran, Iran. Department of computing, imperial college London, Uk. Http://arxiv.org/pdf/1507.08217v1.pdf

[3] Lewis, J., Fowler, M. (2014). Microservices. Chicago, estados unidos. Martin Fowler. http://martinfowler.com/articles/microservices.html

[4] Fátima, Casaú Pérez (2016). Arquitecturas basadas en microservicios. https://comunidad.iebschool.com/fatimacasau/2016/05/18/microservicios-nuevo-paradigma-en-el-desarrollo-de-software/

[5] Esaú, A. (2016). Qué son microservicios y ejemplos reales de uso. https://openwebinars.net/blog/microservicios-que-son/

[6] Chakray Consulting s.l. (2017). qué-son-los-microservicios-definición-características-y-ventajas-y-desventajas. https://www.chakray.com/que-son-los-microservicios-definicion-caracteristicas-y-ventajas-y-desventajas/

[7] José Manuel de paz estrada, tesis de grado, 2017, diseño e implementación de una arquitectura escalable basada en microservicios para un sistema de gestión de aprendizaje con características de red social

[8] Fernando, Arconada, Ostegui, (2016), comunicación http de microservicios

[9] Watson, w. R., Watson, s. L. (2007). An argument for clarity: what are learning management systems, what are they not, and what should they become?. Techtrends (pp. 28-34). Universidad de indiana, universidad purdue, indiana, estados unidos.

[10] Neumann, Sam (2015), building microservices, designing fine.grained systems. Sebastopol, o'reily media.

[11] Tihomirovs, J., Grabis, J.(2016).Comparison of soap and rest based web services using software evaluation metrics. Information technology and management science, 19(1).

[12] Villamizar, M., Garces, o., Castro, H., Verano, M., Salamanca, l., Casallas, y Gil, (2015).Evaluating the monolithic and the microservices architecture pattern to deploy web applications in the cloud. Computing Colombian conference (10ccc), (pp. 583–590).

[13] Tihomirovs, J., Grabis, J (2016). Comparison of soap and rest based web services using software evaluation metrics. Information technology and management science, 19(1).

[14] Wolff, E.(2016). Microservices: flexible software architectures. Createspace independent publishing platform.

[15] Posta, Cristian. (2016).Microservices for java developers - a hands-on introduction to framework & containers. California,, USA : o'reilly media inc., ISBN: 978-1-491-96308-1.

[16] Mike Wasson (2015), estilo de arquitectura de microservicios, content developer at Microsoft. https://docs.microsoft.com/es- Gomez, D. U. Microservicios, arquitectura de información para un desarrollo ágil. https://azure/architecture/guide/architecture-styles/microservices.

[17] Babar, M., Brown, A., Mistrík, I, (2013(. Agile software architecture: aligning agile processes and software architectures.

[18] Leonard Richardson and Sam Ruby. (2007) Restful web services. O'reilly. Beijing Cambridge farnham köln Sebastopol Tokyo. Published by o'reilly media, inc., p. 215

[19] Santiago Pavón Gómez, Manuel Pérez, Herrera. (2015). Cuadrillero. Arquitectura basada en microservicios. [universidad politécnica de Madrid]. Madrid. p. 9.

[20] Arquitectura de microservicios es una arquitectura de software ágil. (2017). http://www.javaworld.com/article/3075880/applicationdevelopment/microservicearchitecture-is-agile-software-architecture.html

21] Daya, S., Van Duy, N., Eati, K., Ferreira, Cc., Glozic D., Gucer, V, Vennam R. (2015). Microservices front theory to practice Ibm Corporation.

[22] Fernando Arcinada Aristegui, (2016), comunicación http de microservicios

[23] Villamizar, M., Garces, o., Castro, H., Verano, M., Salamanca, l., Casallas, y Gil, (2015).Evaluating the monolithic and the microservices architecture pattern to deploy web applications in the cloud. Computing Colombian conference (pp. 583– 590).

[24] Cesare Pautasso. (2011). Soap vs rest. Bringing the web back into web services. IBM Zurich research lab. Http://www.iks.inf.ethz.ch/education/ss07/ws_soa/slides/soapvsrest_eth.pdf

[25] Muehlen, J, Nickerson, Keith. D, Swenson,(2005). Developing web services choreography standards-the case of rest vs. Soap", decision support systems. Vol. 40, no. K, pp. 9-29. ISSN:0167-9236

[26] Curbara, Golan, Kein, Leymann, Roller, Thatte, Weerawarana. (2002). Business process execution language for web services. Version 1.0. (IBM, BEA, Microsoft).

[27] Gomez, D. U. (2015). Microservicios, arquitectura de información para un desarrollo ágil y eficiente. https://pdfs.semanticscholar.org/4372/228c35d06efaa54151168a3f1eff32e2ff0d.pdf

[28] Mauro, T. (2015). Adopting microservices at Netflix: lessons for architectural design. http://repositorio.utp.edu.co/dspace/bitstream/handle/11059/8176/0052g633.pdf?sequence

[29] Jaider, Anillo, Garcia (2017). Orquestación de microservicios. https://docplayer.es/93672987-Orquestacion-de-microservicios-introduccion-a-arquitecturas-de-desarrollo-modernas-basadas-en-sistemas-distribuidos.html